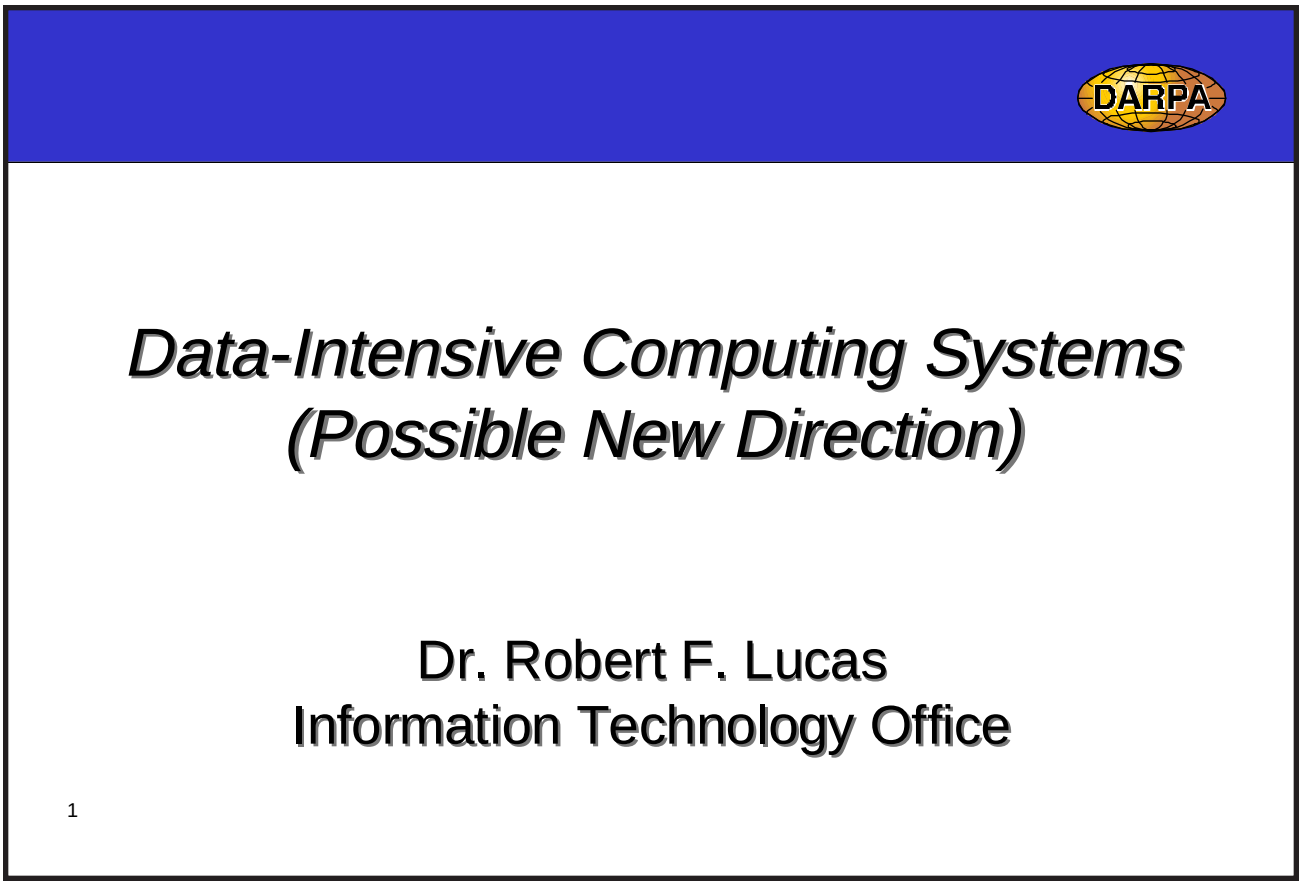
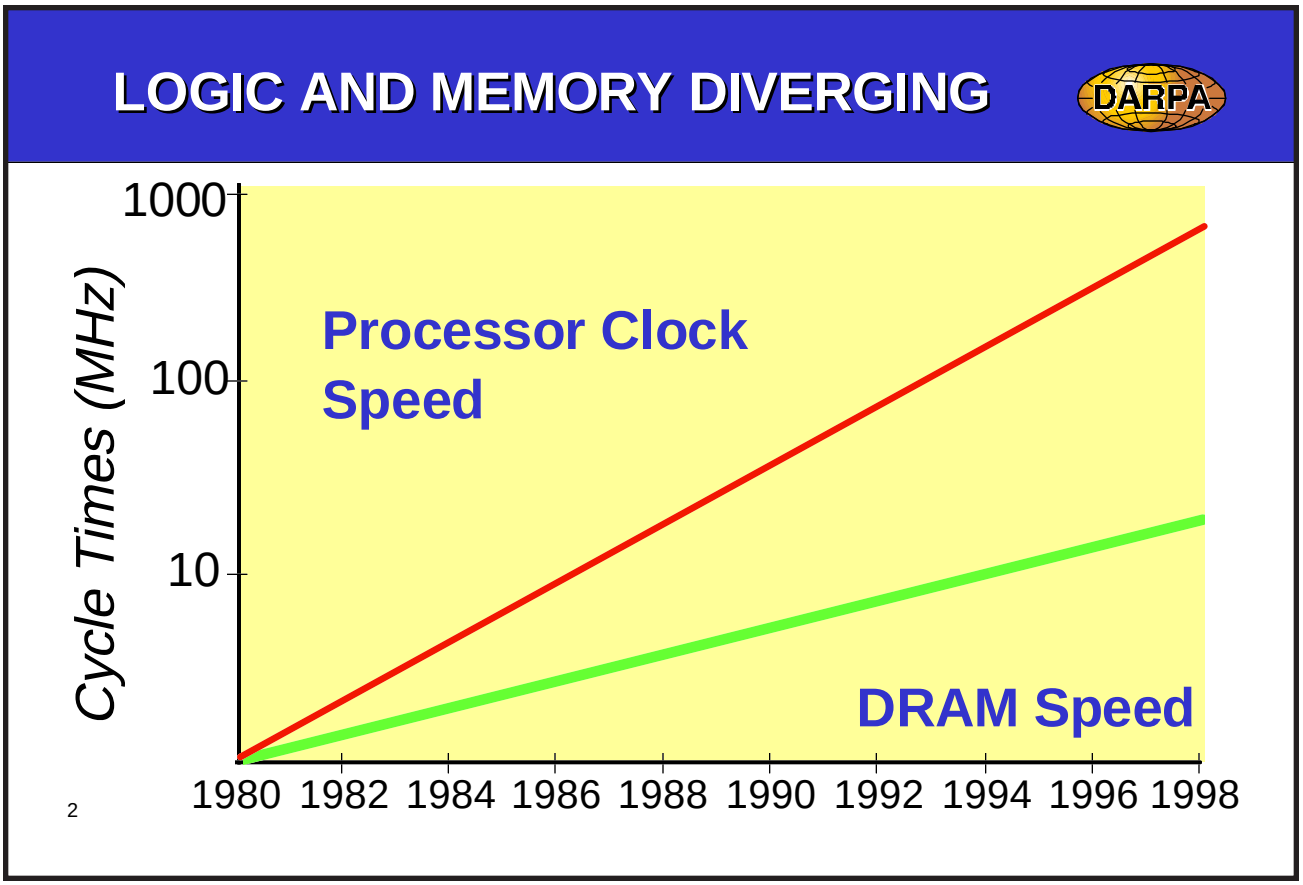


Good afternoon,

It is my pleasure to present to you, Data Intensive Computing, a promising new direction in computer architecture.

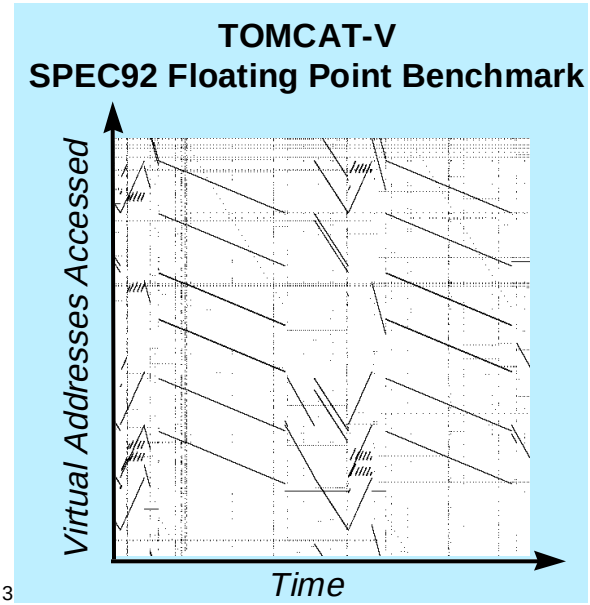
Although Data Intensive Computing is not an approved program and no BAA is planned at this time, I am interested in receiving feedback concerning the concepts I will share with you this afternoon and mechanisms through which they could be realized.

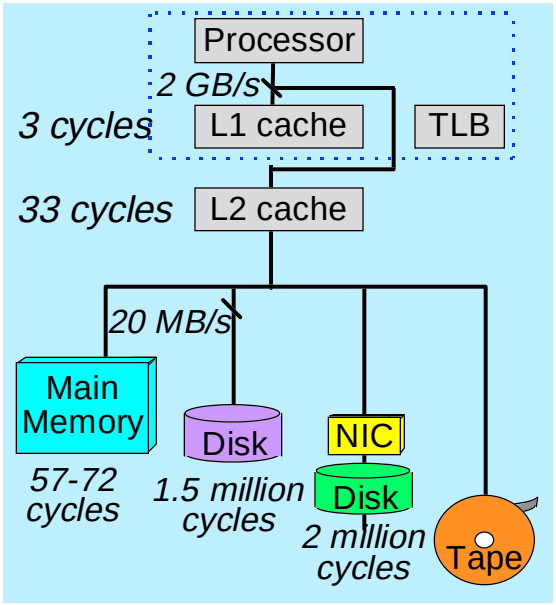


I'd like to begin with a little motivation. This figure displays trend lines representing the relative performance growth of microprocessors and DRAM memories. Microprocessors are doubling in speed every 18 months. DRAMs, on the other hand, are increasing in speed by only 7% per year.

VIRTUAL MEMORY HIERARCHY

Premise: *Applications Have Small Working Sets of Contiguous Data*



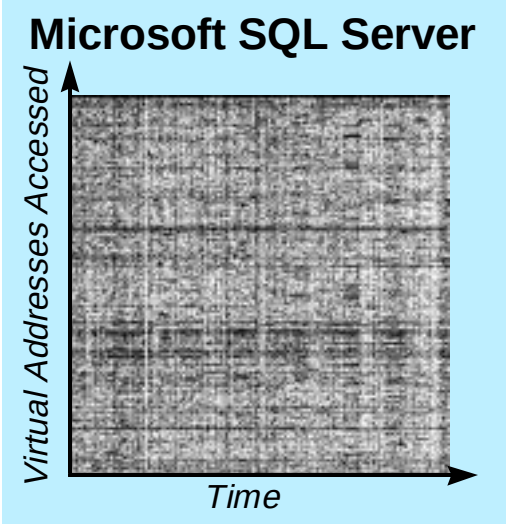


3

How have computer architects responded to this problem? On the left we have a trace of the memory accessed by a commonly used benchmark. Notice that at any point in time, only a small subset of the memory is being accessed. Furthermore, the program advances through its data at a stride of one. Therefore, a hierarchy of small, fast, cache memories provides the user with the illusion of a large, flat main memory.

DATA-STARVED DEFENSE APPLICATIONS

Many Defense Applications Have Large Data Sets That Are Accessed Non-Contiguously



Data Starved Applications

- Radar Cross-Section Modeling (FMM)
- High-Definition Imaging
- Terrain Masking
- Relational Databases
- Object-Oriented Databases
- Structural Dynamics
- Circuit Simulation

4

Unfortunately, many important Defense applications have neither small working sets nor contiguous access patterns; the same is true of a number of commercial object-oriented systems and databases. Consider the example on the left. This is the trace of the memory access pattern generated by a relational database. On the right is a list of applications critical to Defense that are also starved for data by the modern virtual memory hierarchy and thus perform at a small fraction of the potential power of their RISC hosts.

MISSION CRITICAL APPLICATION SOLUTIONS

Traditional:

	Advantages	*Disadvantages*
• Multiple COTS CPUs	• Relatively inexpensive • Preserves S/W investment • Low risk integration	• Slow • Big
• Special Purpose Devices (SPDs)	• Fast • Small	• Expensive • Maintenance liability • Abandons software investment

New Program:

Data Intensive Computing System: Integrate processing and communication into memory image of standard CPU

5

If you need to increase the memory bandwidth delivered to your application, what are the alternatives?

You can buy multiple COTS CPUs if the application scales. However, the resulting systems tend to be large and slow.

Alternatively, you can produce a custom solution at great cost in both hardware and software. This becomes a long-term maintenance liability.

We propose a new approach that allows the application to directly control its memory and communication assets.

DATA INTENSIVE GOAL

New Memory Architecture to Enable Defense Applications to Overcome Limitations of the VM Hierarchy.

- Incorporate it into a new computing system.
- Demonstrate its effectiveness.

Metric: Two Orders-of-Magnitude Faster Than Contemporary VM Systems of Comparable Cost.

6

The goal of Data Intensive Computing is to develop a new memory architecture that enables important Defense applications to overcome the limitations of the virtual memory hierarchy. It should be possible to demonstrate that these applications will run two orders-of-magnitude faster on the new architecture than on contemporary virtual memory systems.

WHY NOW?

Rapidly Growing Need

- Memory bottleneck severely constrains performance
- Increasing mismatch in processing/memory speed
- Bandwidth per bit is plummeting

Opportunity Arising

- Integrate processor & DRAM on same die
- Fertile ground as CPU and system research converging
- Memory dominates cost of system (~90%)

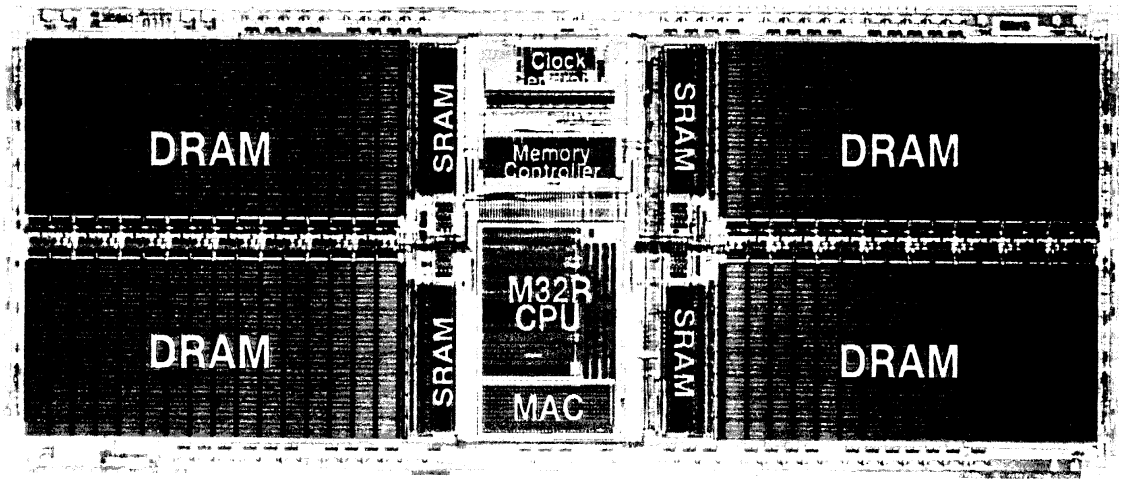
7

This is the perfect time to address this problem. We've already discussed the increasing mismatch in processing and memory speeds. This is creating a severe constraint on performance. In fact, DRAM density increases faster than I/O bandwidth, reducing the bandwidth per bit delivered to an application.

As the magnitude of the problem grows, so does the opportunity. It is now possible to integrate processors and DRAM on the same die. Furthermore, in high-end workstations, memory dominates the total cost already, so this is where the leverage is.

EXAMPLE OF PROCESSOR AND MEMORY CHIP

M32R/DRAM



MITSUBISHI ELECTRIC

8

Here is an example of a processor fabricated on the same die as its DRAM memory. IBM and other memory vendors also have the technology to do this.

TECHNICAL CHALLENGES

Maximizing Memory Bandwidth

Processors within memory manipulate data

Optimizing Data Movement

Applications manage memory hierarchy so data movement is tailored to specific needs

Split Basing

Integrate network and processor architecture minimizing access time to remote data

9

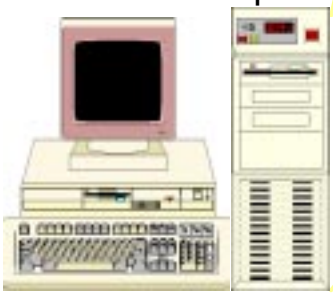
We see at least three opportunities for technical innovation:

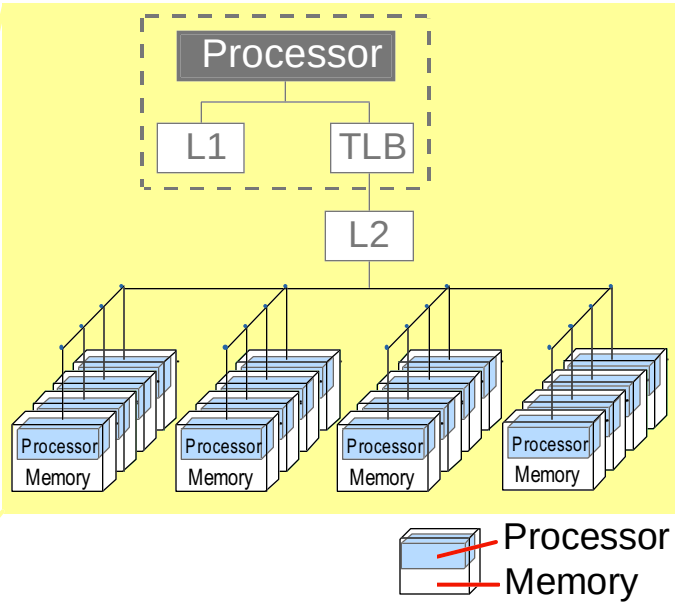
1. Maximize the data bandwidth delivered to an application. This can imply moving at least part of the application to its data so that it can be processed.
2. Allow applications to choreograph the movements of data within the memory hierarchy, so that their specific needs are met.
3. For applications that involve multiple computers, minimize the latency to access remote data.

MAXIMIZE MEMORY BANDWIDTH

Put Processing in DRAM Chips

- Execute object oriented methods at the site of the object's data
- Search and sort in parallel
- Automatic garbage collection and compacting



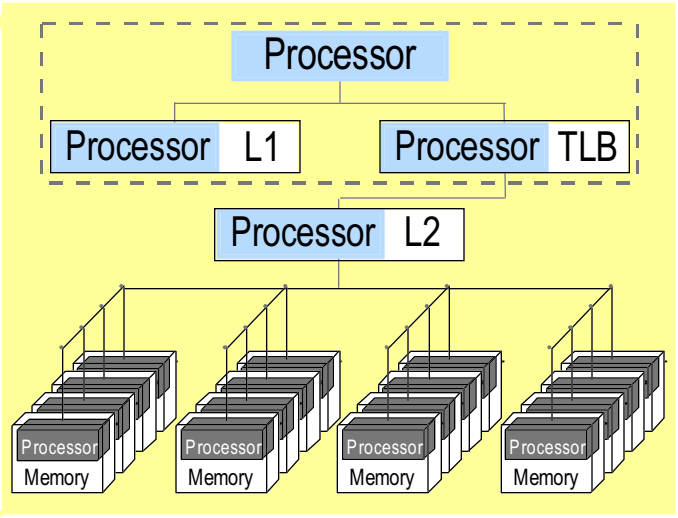


10

This picture represents an approach to increase the effective memory bandwidth delivered to an application. Processors are fabricated on the same die as the DRAM that comprises the main memory of the system. Application instructions can then either execute in the host CPU, execute in the memory, or both.

APPLICATION MANAGEMENT OF MEMORY HIERARCHY

Allow Applications to Manage the Memory Hierarchy So Data Movement Is Tailored to Their Specific Needs

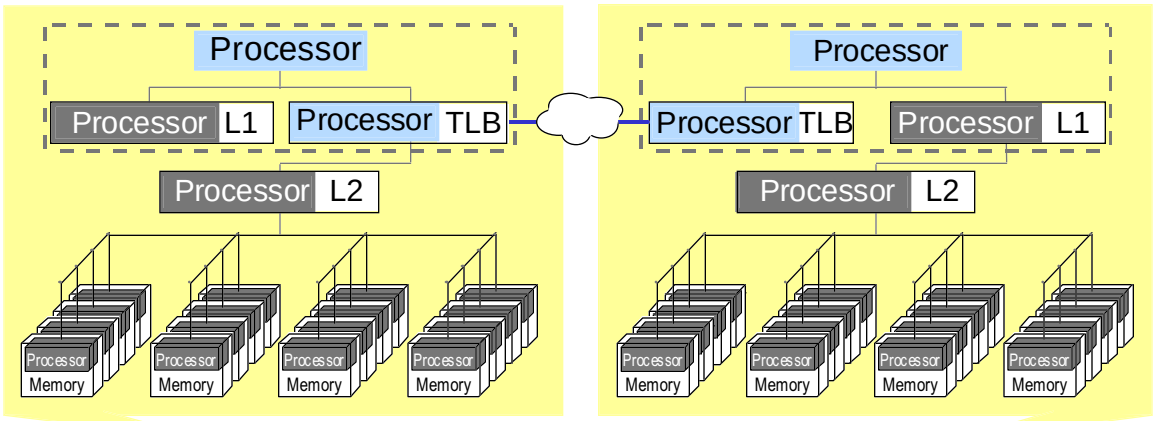


- Optimize data movement
- Increase cache utilization and effective memory bandwidth

11

Many Defense applications have non-unit or even pseudo-random memory access patterns. These make cache a liability in that long cache lines and memory bandwidth are grossly under utilized. We could improve this by allowing applications to dictate the placement and movement of data throughout the memory hierarchy.

SPLIT-BASING ARCHITECTURE



Integrate Remote Network Access Into the Processor Architecture

- Eliminate software overheads to access remote data
- Fetch remote data in under 1µsec

12

RESULTS QUANTIFIED

	HP PA8000 ($64K)	*Data Intensive Computing System (~$80K)*	*Performance Improvement*	*Cost to Match*
Bandwidth				
Contiguous to Application	282 MB/s	61400 MB/s	218X	$13,935K
Random to CPU	70 MB/s	1440 MB/s	21X	$1,317K
Random at Memory	—	3840 MB/s	55X	$3,511K

13

To better understand the impact of the program, this table compares an existing workstation to one enhanced with Data-Intensive technology. The new system is assumed to have 16 “processor in memory” chips on each of four memory banks.

RESULTS QUANTIFIED (cont.)

	HP PA8000 ($64K)	*Data Intensive Computing System (~$80K)*	*Performance Improvement*	*Cost to Match*
Compress 1GB	3.6 sec	0.016 sec	225X	$13,935K
OODB Ad-Hoc Query	1.8 sec	0.008 sec	225X	$13,935K
Sparse Ax=y	8.3 MFlops	120 MFlops	15X	$960K

14

Here we examine the impact on some common applications:

- Many applications leverage dynamic management of storage that has to be compressed periodically.
- In OODBs one is often confronted with the need to search for data that cannot be found on the index trees.
- Sparse matrix multiplication is the key to many large scientific and engineering applications.

WHAT IS ENABLED?

- Searching Large Object-Oriented Databases Will Be Two Orders-of-Magnitude Faster
- Object Oriented "Methods" Performed in Memory, With Lower Latency and Higher Bandwidth Access to Their Data
- Utilization of High-Speed Memory Will Increase 16-Fold for Large Applications With Non-Contiguous Data Access Patterns

15

Data-Intensive Computing could enable a substantial increase in performance of data-starved applications. Object-oriented databases are a prime example. By searching and sorting without the need to move pointers and indices all the way up the memory hierarchy, we could achieve orders-of-magnitude in improvement. Object-oriented methods could execute in-place, reducing memory traffic and dramatically increasing performance. In addition, allowing applications to choreograph data movement will maximize the utilization of precious cache and memory bandwidth.